Design of High-Accuracy Multiple Flyby Trajectories Using Constrained Optimization

Dennis v. Byrnes*
Larry E. Bright†

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, Ca 91109

*Member Technical Staff, Member AIAA, Member AAS
†Member Technical Staff

INTRODUCTION

Over the past two decades the extremely difficult problem of Efficiently finding optimal trajectories which involve close flybys of multi ple gravitating' bodies has been addressed by many investigators using a variety of met] iods. Whereas the method and particular formulation of the method described in this paper are in part directly related to several of those methods, a novel way of splitting the trajectory into legs and the! continued improvement of computing capability of modern computer workstations has allowed for a significant improvement in sped, ease of USC, and accuracy of the resulting trajectory design.

The trajectory optimization technique described in this paper provides several distinct advantages over previous formulations. First, fully numerically integrated trajectory modeling is used. That is , no approximations to the trajectory arc made and the inclusion of any level of complicated force models desired is allowed. Second, only trajectory propagation is used so there is no requirement for solutions of multiple boundary value problems as i ntermediate steps before optimization. This is accomplished by the novel method of splitting the trajectory into independent legs, which arc then subjected to constrained optimization. Third, each of the trajectory legs may be specified by any convenient set of parameters particularly useful for that leg. Any of these parameters may then be subject to constraints. Fourth, the nonlinear optimization problem is solved by solving a sequence of linear problems which converges to the optimal nonlinear solution. Fifth, the robustness of this formulation requires 1 ittle or no user interaction with the optimization once a feasible problem has been pod. Finally, although it is not the subject of this paper, our software implementation of" this method makes use of' Fortran 90, modern techniques of object oriented programming, and extremely fast UNIX based workstation computers.

Optimal trajectories are determined starting from a specified state vector or launch conditions, consistent with a specified set of constraints and a specified flyby body sequence, and, optionally, in the case of interplanetary t rajectories, meeting certain arrival conditions. Trajectory modeling is based on numerical i ntegration of' the equations of motion of a point-mass spacecraft subject to gravitational accelerations. Additional] effects modeled are impulsive and/or finite motor burns and solar])rc.s.sure. The gravitational models i nclude, in addition to the invcrsc-

square acceleration due to the planetary central body: point-mass gravitational accelerations due to any combination of sun, planets and satellites plus acceleration due to the oblateness of planetary central bodies, Trajectory constraints may include flyby altitudes, b-plane angles, latitudes, times of closest approach to flyby bodies, inclinations or essentially any other orbital parameter with respect to either the primary or secondary body. There may also be constraints on maneuvers such as time, direction, location and mode of execution. Any of these constraints may be equality, inequality or bounds constraints, and will, of course, be closely related to mission operations requirements and science objectives.

The method of solution used is a parameter optimization algorithm based on a series of linearizations of the "real" highly nonlinear problem. The optimization algorithm changes the independent variables (a series of estimated states along the trajectory, usually at flybys) on successive iterations to reduce the cost function (total ΔV). Due to the complexity of the problem, it may sometimes be necessary for the user to actively control the optimization process in order to achieve convergence. A variety of control procedures are available.

OPTIMIZATION PROBLEM STRUCTURE

Trajectory Structure

A complete trajectory is broken up into a sequence of user-defined trajectory legs. The legs are contiguous in time. The boundary between two successive trajectory legs is referred to as a trajectory breakpoint. On each trajectory leg there is a uniquely distinguished point referred to as the control point for that leg. A set of six control point variables is defined at each control point by the user. The set is chosen from a wide range of possibilities separately for each trajectory leg.

The control point variables collectively, over all legs of the trajectory, determine the initial conditions for trajectory propagation and are a subset, of the independent variables for optimization. Any leg Of the trajectory may optionally have an additional three independent variables which are associated with a maneuver at the beginning of the trajectory leg. These ΔV variables may be included in the independent variable set in order to impose constraints on them. Finally, besides the control point variables and the optional ΔV variables, the

independent variables include the times of the breakpoints between trajectory legs. Since maneuvers always occur at the beginning of trajectory legs, this permits optimization of maneuvers times if these variables are free parameters.

1 Breakpoints and control points occur in alternating fashion along a trajectory. For example, on a trajectory with n legs, they occur in the following order:

B0, C1, B1, C2, ..., Cn, Bn,

where the Bi's are breakpoints and the Ci's are control points. The entire trajectory begins at the initial breakpoint B0 and ends at the terminal breakpoint, Bn.

Trajectory Generation

On a given iteration of the optimization procedure, a tri a] trajectory must be generated from the current values of the independent variables. These legs of a trajectory are generated separately and independently. Each leg is the result of two trajectory propagations: first, a reverse (i .e., backwards in time) propagation from the control point for the leg to the epoch of the starting breakpoint of the leg; second, a forward propagation from the control point to the ending breakpoint of the leg. The trajectory generation process is complete when all the trajectory legs have been determined in this way. Although the current implementation of the software processes the legs sequentially, since the legs are i ndependent this formulation is ideal for possible implementation using parallel processors.

At each trajectory breakpoint there is an "incoming" velocity, which results from forward propagation of the trajectory from the preceding control point, and an "outgoing" velocity, which results from reverse pro] agation of the trajectory from the succeeding control point. 'Thus, while the prop agation procedure guarantees that a trajectory is continuous within a leg, there will in general be discontinuities in both position and velocity at the breakpoints between legs. In order to achieve a final trajectory that is continuous in position, constraints are automatically imposed on the optimization requiring that position discontinuities at breakpoints be zero. Velocity discontinuities, on the other hand, cannot always be eliminated, since maneuvers may be physically necessary in order to fly the trajectory within the

constraints. Within the! available degrees of freedom, however, the user may choose to impose! constraints requiring that velocity discontinuities at selected breakpoints be reduced to zero.

Placement of Breakpoints and Control Points

For reasons that should be clear from the preceding discussion of trajectory generation, maneuvers that are subject to optimization may only occur at trajectory breakpoints. Thus, breakpoints will normally be placed at points on the! trajectory where AV'S are required or expected to be needed. Apart from this consideration, the placement of trajectory breakpoints is arbitrary. As indicated above, the control point for a leg occurs between the epochs defining the bounding breakpoints of the leg. Otherwise, control points may be placed arbitrarily by the user. In fact, a control point and a breakpoint may coincide.

A trajectory leg may contain zero, one or more! flybys of Gravitating bodies. In normal practice, trajectory breakpoints will be at or near the expected maneuver epochs, with at least one between significant gravity-assist flybys. A leg will typically contain one flyby; the control point for the leg will typically be placed at or near the periapsis of the flyby. None of these situations is in fact required by the method however. Note! that non-zero maneuvers will in general occur at the initial breakpoint, B0, and at the terminal breakpoint, Bn, if not otherwise constrained.

ITERATIVE SOLUTION OF THE OPTIMIZATION PROBLEM

The optimization problem is a highly nonlinear one due to the nonlinear nature of the multi-body equations of motion. The nonlinear problem is solved as the! limiting case of a series of linear (or "linearized") problems. Each linearized problem is itself solved iteratively as a sequence of problems through a process called "re-weighting the cost function".

Thus, the method consists of a loop-within-a-loop structure, with the inner iteration controlling the re-weighting process and the outer iteration controlling the re-linearization process. When both iterations have converged, the "real" nonlinear problem with the "real" sum-of-magnitudes cost function has been solved.

The cost function that is minimized is the sum of the AV magnitudes. For numerical reasons, an indirect approach to computing this cost function is used. The cost function is:

$$\Delta V_{total} = w_1 \Delta V_1^2 + W2AV2^2 \text{-t} \ldots + w_n \Delta V_n^2$$

where:
n = number of allowed maneuvers on the trajectory
ΔV_i = magnitude of the ith AV vector
and Wi = scale factor or "weight" associated with t] ie ith maneuver

The wi 's are initially set to the somewhat arbitrary value of 1.0. They are then adjusted in a series of automated "re-weighting" operations. When iterative rc-weighting is finished, each weight, equals

1/ (the corresponding AV)

or, expressed differently, the reciprocal of the weight equals the corresponding ΔV nagnitude. The cost function, then, reduces to precisely

$$\Delta V_1 + AV2 \text{-t} \ldots + \Delta V_n$$

e., the sum of the AV magnitudes, which is the "real" cost function to be minimized.

For each set of weights, the current linearized version of the trajectory optimization problem is solved by a utility linear-least-squares minimization package. When the re-weighting process converges, the resulting trajectory so] ution is used to determine a new linearization of the nonlinear problem; the re-weighting process is done on this new linearization, etc., until there is only a negligible change from one linearization to the next.

RESULTS AND EXAMPLE TRAJECTORIES

Descriptions of the convergence behavior and solution characteristics for a selection of different trajectory types are presented. At least several cases from the extensive set of baseline test cases used to validate the current software implementation will be summarized. They may include some of the following:

Galileo Interplanetary VEEGA Trajectory
Earth-Venus-Earth-Gaspra-Earth-Ida-Jupiter

Galileo Satellite Tour
11 orbits of Jupiter with 1 or 2 satellite flybys per orbit
2 of the orbits with constrained Jupiter inclination

Cassini interplanetary VVEJGA Trajectory
Earth-Venus-Venus-Earth-Jupiter-Saturn

Cassini Satellite Tour (Example)
approximately 40 orbits of Saturn with 0,1 or 2 flybys per orbit
several orbits with constrained Saturn inclination]]

Near Earth Asteroid Rendezvous Trajectory (NEAR)
Earth-Earth-Eros

Multiple Lunar Flyby Trajectory

The samples will be chosen to demonstrate the versatility of the method and its application to a wide variety of current and proposed multiple flyby missions.